

**REMARKS**

Claims 1-32 are pending in the present application, with claims 1, 12, 22, 26, and 30 being the independent claims. In the outstanding Office Action, claims 10-32 are rejected under 35 U.S.C. § 101, claims 28-29 are rejected under 35 U.S.C. § 112, ¶ 2, and claims 1-32 are rejected under 35 U.S.C. § 102(e) as anticipated by U.S. Patent No. 6,308,274 (Swift).

At the outset, Applicants respectfully ask that the Examiner acknowledge the priority claim of this application to U.S. Provisional Application No. 60/214,811 filed on June 28, 2000.

***Rejection of Claims 10, 11, 12-29, and 30-32 under 35 U.S.C. § 101***

Regarding claim 10, Applicants have added the limitations “implemented by a computer” and “tangible” to specify where the instructions are carried out and in what type of medium they are located, respectively. Regarding claims 12-29, Applicants have amended the claims to additionally recite “A tangible computer readable medium” instead of the original “A computer readable medium.” Regarding claims 30-32, Applicants have (1) amended the claim to highlight the functional language and (2) added the recitation “a tangible computer readable medium” instead of the original “a computer readable medium.” Lastly, Applicants have canceled claim 11.

Applicants note that none of the amendments entered herein are meant to limit the scope of the claims in any way. Rather, these amendment are made to bring out inherent characteristics and thus to more clearly recite the pending claims.

***Rejection of Claims 28 and 29 under 35 U.S.C. § 112, ¶ 2***

Applicants have amended claims 28 and 29 in order to address lack of antecedent basis in those claims. Specifically, Applicants have amended “said dynamic access element” to “a dynamic access element” in both claims.

***Rejection of Claims 10 and 12-29 under 35 U.S.C. § 102(e)***

First, as mentioned above, claims 1, 12, 22, 26, and 30 are the independent claims. For example, claim 1 recites:

A method for dynamically managing access to a resource in a computer system, the system having a client thereof making an access request for the resource, the method comprising:

determining, via an application programming interface, based upon dynamic data and first *dynamic policy* whether a client authorization context is to be updated, wherein said first *dynamic policy* is tailored to an application through which the resource is accessed;  
identifying an access control entry as a callback access control entry; and  
in response to identifying the access control entry as a callback access control entry, evaluating, via said application programming interface, based upon dynamic data and second dynamic policy whether said callback access control entry bears on said access request, wherein said second dynamic policy is tailored to said application.

(emphasis added). Applicants submit that the above emphasized notion of “determining...based upon ... *dynamic policy*...” is distinguishable from what is taught by Swift.

Swift, discloses a method and mechanism to enforce reduced access via restricted access tokens. Restricted access tokens are based on an existing token, and have less access than that existing token. A process is associated with a restricted token, and when the restricted process attempts to perform an action on a resource, a security mechanism compares the access token information with security information associated with the resource to grant or deny access. *See* Abstract and col. 6, l. 5 to col. 7, l.35. However, Swift is silent on employing such *dynamic policy* (especially a dynamic policy that is tailored to an application through which the resource is accessed).

Independent claims 12 and 22 recite similar limitations, and are therefore deemed patentable for similar reasons: “determining, via an application programming interface, based upon dynamic data and dynamic policy whether said client authorization context is to be updated, wherein said dynamic policy is tailored to an application through which the resource is accessed” (claim 12); and “in response to identifying the access control entry as a callback access control entry, determining, via an application programming interface, based upon dynamic data and dynamic policy whether said callback access control entry bears on said access request, wherein said dynamic policy is tailored to said application” (claim 22).

Second, independent claim 26 recites:

For an application in a computer system having a resource manager that manages and controls access to a resource, a tangible computer readable medium bearing computer executable instruction for carrying out a *dynamic authorization callback mechanism*

that provides extensible support for application-defined business rules via a set of APIs and DACLs including a dynamic groups element, which enables an application to assign temporary group membership, based on dynamic factors, to a client for the purpose of checking access rights.

(emphasis added). As the above claim explains, “*a dynamic authorization callback mechanism* ... provides extensible support for application-defined business rules via a set of APIs and DACLs including a dynamic groups element.” Swift mentions “a security mechanism” but falls short of providing the kind of mechanism that is recited above.

Lastly, independent claim 30 recites:

A data structure stored on a tangible computer readable medium for use in connection with dynamic access check determinations for an application in a computer system, the data structure comprising:  
an identifier for identifying the data structure as *a callback access control entry*; and  
dynamic authorization policy data in a format tailored to the application to handle access requests.

(emphasis added). Applicants submit that Swift is silent on a “callback access control entry.”

In fact, the specification further explains the idea behind this limitation:

Dynamic Access Check enables an application to perform customized procedures for checking access rights, also based upon transient or changing factors such as data from client operation parameters, authorization policy data stored in *a specialized callback Access Control Entry (ACE) type designed for dynamic access check policies*, and any other authorization policy data that may be managed, computed or retrieved by the application, service or object.

(Specification, p. 4, l. 27 to p. 5, l. 3) (emphasis added). Swift is silent regarding a “callback access control entry” for “dynamic access check policies.” Applicants note that they cannot prove a negative, i.e., what is *not* in Swift. Therefore, the Applicants respectfully ask that the Examiner point out which specific components in Swift correspond to those limitations recited and emphasized above.

Inasmuch claims 2-11, 13-21, 23-25, 27-29, and 31-32 depend either directly or indirectly from independent claims 1, 12, 22, 26, and 30, respectively, they are believed allowable for the same reasons. Withdrawal of the rejection under § 102(e) is therefore earnestly solicited.

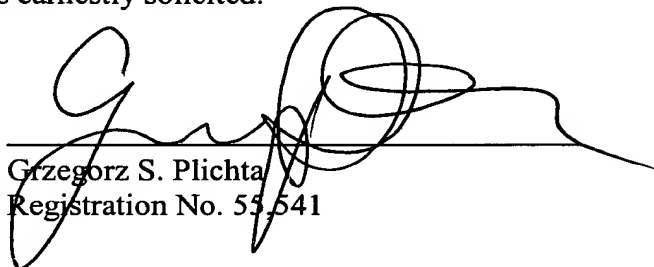
**DOCKET NO.:** MSFT-0222 / 158379.02  
**Application No.:** 09/849,093  
**Office Action Dated:** July 1, 2005

**PATENT**

### **CONCLUSION**

Applicants believe that the present Amendment is responsive to each of the points raised by the Examiner in the Official action, and submits that Claims 1-32 of the application are in condition for allowance. Favorable consideration and passage to issue of the application at the Examiner's earliest convenience is earnestly solicited.

Date: September 30, 2005



Grzegorz S. Plichta  
Registration No. 55,541

Woodcock Washburn LLP  
One Liberty Place - 46th Floor  
Philadelphia PA 19103  
Telephone: (215) 568-3100  
Facsimile: (215) 568-3439